Brief in Support of Appeal

Inventor: Name: David Ge
Phone: (425)803-0679
Email: dge893@gmail.com

Appeal: Final rejection to application 09/682,315

Summary of Invention:

[0014] In general, in one aspect the invention is a visual programming tool and method that includes an action object and an action list object, and a set of pre-developed objects which are derived from a generic object which supports a property-event-method model. The action list object contains a sorted list of action objects. An action object has, as its members, an action performer which is a derived non-abstract object of the said generic object; an action method which is one of the methods supported by the action performer; action data which are the parameters needed by the action method. The action list object is created and used by the user as event handlers.

[0015] The generic object is an object from which the other pre-developed objects are derived. Thus, it contains a set of properties and methods that are shared by the other pre-developed objects in the tool. The generic object makes a generic programming model and execution environment possible while more and more pre-developed objects may be added to the said programming tool in the future.

[0016] In general, in another aspect, the invention is a visual programming method that is implemented on a computer having a persistent storage, a display screen and one or more input devices which a user employs to command the said programming tool to develop software applications. The method includes: the steps of defining and supporting by computer implemented steps a set of pre-developed object classes, an action object class and an action list object class; in response to input from the user, creating object class instances from the said pre-developed object classes and graphically presenting the said object class instances on the display screen; in response to input from the user, instances of object classes being manipulated graphically to form visual presentations of the software application; in response to input from the user, each property of each instance of object class being set; in response to input from the user, selecting an event of an object class instance and assigning an action list object as the event handler; in response to input from the user, each said action object in the said action list object being created by steps of 1) picking an instance of object class as the action performer from the existing instances of object classes presented to the user in an organized manner; 2) picking a method as the action method from the supported methods of the said picked instance of object class, the said methods are presented to the user on the display screen for the user to pick; 3) picking/specifying data for the said picked method via one or more dialog-boxes, if action data is required for the said picked action method; indicating, in response from the user input, which object class instances are initially active objects; saving said object class instances and said action lists, together with the relationship between action lists and events of the said object class instances, and the indications of which objects are initially active, to the computer persistent storage.

[0017] In general, in another aspect, the invention is a software execution environment, which reads back from the computer persistent storage the said saved object class instances, action lists and the relationship between action lists and events of the object class instances, and the indication of which objects are initially active; creates and displays the object class instances which are initially active; responses to each event by sequentially executing each action in the said action list object assigned to the said event by the following steps 1) locating the object class instance which is assigned as the action performer for the action; 2) signaling to the said action performer which action method is specified for the action; 3) if the method data is specified for the said method of the said located object class instance, the method data is passed to the said object class instance as well; 4) the said located object class instance carries out the said action method.

[0018] The invention provides an intuitive graphical interface that enables non-technically oriented users to easily develop computer software applications, without any computer languages involved, not event script languages. It will open a new door of computer programming to a much wider range of people.

Issues

The major issues of the rejection are related to my added words "in a codeless manner without using a compiler" in an amended claim. These words were considered by the examiner to be 1) a new matter; 2) not enabled by my claims. This is actually not a new matter because the steps in my claims were not changed. My amendments mostly were to re-order the original steps. The examiner failed to recognize that the steps in my claims put together realize a codeless programming system.

Other issues are caused by the examiner's misunderstanding the contents of his own references the examiner cited; and caused by "off target" rejections. By "off target" I mean words in my claims were taken out of context for rejections.

Group of Claims

Claim 1, claim 2, claim 3 and claim 7 form a codeless programming process which let the user create computer programs visually without using computer languages (thus it is codeless). Claim 6 describes the process of program execution, which shows that the programming results are executed directly without compiling (thus a compiler is not needed).

Arguments

Argument 1.

Regarding office action page 2, item 1 and item 2

Rejection reasons to be argued: "creating an application program in a codeless manner without using a compiler" is a new matter and is not enabled by the disclosure.

Argument details:

My words, "in a codeless manner without using a compiler", are a statement of fact. The examiner linked these words to the following words in step A of my claim: "defining and supporting a set of pre-developed object classes", and hastily got conclusion that my

statement was not supported clearly by the disclosure. The examiner failed to see that the statement is supported completely by putting claims 1, 2, 3, and 6 together.
Claim 1, 2, and 3 describe an interactive process to let the user (as a programmer) create an object instance, an action list, and a mapping of action list to an event. As the steps shown, the whole process is completed by presenting selections to the user and let the user pick the choices, there is not coding or computer languages involved. Thus my statement of "in a codeless manner" is fully supported.
In software engineering, compilation means transferring contents from one format(s), usually in plain text, to another format(s), usually in binary executable format. Claim 6 describes the process of saving, loading and executing the programming results done by the user. As shown in steps F, G, H, H1, H2, H3, H3a, H3b, H3c, H3c1, H3c2, H3c3, and H3c4, the execution is done using the objects created in the process described in claim 1, 2, and 3. Therefore there is not a content format conversion involved. In another word, there is not a compiling involved. Thus my statement of "without using a compiler" is fully supported.
Since all these steps exist in my original patent application, my statement of "creating an application program in a codeless manner without using a compiler" is NOT a new matter. Thus the rejection reasons are groundless.

Argument 2.
Regarding office action page 2, item 3, page 3, item 4
Rejection reasons to be argued: a) "sorting a list of items to display"; b) "creating an application program in a codeless manner without using a compiler" is a new matter and is not enabled by the disclosure.
Argument details:
Rejection reason b) is completely identical to item 2. Therefore Argument 1 above applies to it.
Rejection reason a) is an "off target" rejection. I am not patenting "sorting a list of items to display". If I were to patent "sorting" then I would have had to present sorting algorithm. In my claims I only mentioned "the action list class contains a sorted list of action class instances" (Step A). I am patenting an application creation process which uses action class and action list class to realize codeless programming. Suppose a patent application claims that "a car with 4 wheels which can run on road and can fly", if an examiner rejects the claim on the base that "a car with 4 wheels" already exists then that is an "out of context" rejection, like we see here rejection based on using sorted list.
Also, my action list is not for displaying items as the examiner thinks. It is for executing actions in the user specified order as stated in claim 6: "H3b. If the said mapping relationship exists, sequentially performing each action in the said action list mapped to the said event". The examiner assumed my action list is for displaying. But in nowhere in my claims it is so stated.
We can see that the examiner takes my words out of context for rejection and also puts something not in my claims for rejection.

Argument 3.
Regarding office action page 3 last paragraph, page 4, and page 5 the first paragraph

Rejection reasons to be argued: a) claim for "persistent storage" is not patentable; b) "creating an application program in a codeless manner without using a compiler" is a new matter and is not enabled by the disclosure; c) reference quoted by the examiner already describes codeless programming without compiling.

Argument details:

Rejection reason a) is another example of "off target" rejection. I am not patenting "persistent storage".

Rejection reason b) is completely identical to item 2. Therefore Argument 1 above applies to it.

Rejection reason c) is from the examiner's misunderstanding of what he is quoting from his own references. The examiner asserts that Patel pages 265 – 268 describe codeless programming without compiling and compiler. The examiner gets his conclusions by simply looking for word "compiling" and not finding it (office action page 4: "Patel makes no reference to compiling"; office action page 5: "again that no reference is made to compiling"). Using such "word-matching" examining method, in Patel page 268, we can find following statement: "The BeanBox will generate and **compile** an adapter class – **code** that will be executed when the actionPerformed event occurs." It should be clear that a compiler is needed, and coding is used. In simplest case of single method call without parameters, the code generated by the BeanBox is enough. Apart from the simplest case, the user (programmer) needs to do coding by hand. See Patel page 277: "The user can then enter an item **code** to query and press the Price Inquiry button." We can see that even using such "word-matching" examine method, the examiner omitted the clear statements in Patel's teaching, which are in favor of my claims.

Even using such "word-matching" examine method, it still should be based on common sense in Software Engineering. Java code needs compiling. This is a common sense. Therefore, if Patel does not mention compiling then the conclusion should be that a compiler is needed, unless Patel specifically says that it is a completely codeless system without using compilers. The examiner got the wrong conclusion, showing a fundamental misunderstanding of what Patel is teaching.

I would like to give a brief description of the state-of-art in this specific area as a background for my invention, although such background is known to anyone who has knowledge of computer programming. There are many Visual IDE systems (Integrated Development Environment) available since 20 years ago. They let the user interactively create object instances and link actions to events, like Patel described. But they represent actions in code and linking of actions to events is also done in computer languages. The system (Patel) referenced by the examiner also works in this way. In my invention, because of using my action class and action list class, such action making and action linking are done in a codeless manner and without using a compiler. See Argument 1 for details.

Argument 4.

Regarding office action page 5, the second paragraph

Rejection reasons to be argued: claim for "a display screen and one or more input device" is not patentable.
Argument details:
This is another example of "off target" rejection. I am not patenting "a display screen and one or more input device".

Argument 5.
Regarding office action page 5, the third paragraph
Rejection reasons to be argued: claim for "the input device ... to create visual representation..." is not patentable.
Argument details:
This is another example of "off target" rejection. I am not patenting "the input device ... to create visual representation...". Such statements can be found in many patents, for example, in claim 1 of patent **5,862,379**, and in claim 1 of patent **5,850,548**.

Argument 6.
Regarding office action page 5, the last paragraph and page 6, the first paragraph
Rejection reasons to be argued: claim for "an object" is not patentable.
Argument details:
This is another example of "off target" rejection. I am not patenting "an object".

Argument 7.
Regarding office action page 6, the last paragraph, page 7, and page 8 the first paragraph
Rejection reasons to be argued: the action class and action list class in my claims cannot make codeless programming without compiling happen.
Argument details:
What makes my system codeless? The answer is my action class and action list class, and the process of building them and using them. Claim 1, 2, and 3 describe an interactive process to let the user (as a programmer) create an object instance, an action list, and a mapping of action list to an event. As the steps shown, the whole process is completed by presenting selections to the user and let the user pick the choices, there is not coding or computer languages involved in the steps. Thus my system is codeless, and the codeless programming process is clearly defined and taught in my claims.
The examiner says:"The applicant's specification indicates that the user builds the action list (see sect. 0066), which does not appear to be codeless."
I reviewed sect. 0066 in my application. The contents are as following:

[0066] Referring to FIG. 11, Project-Manager lets users create, edit, delete projects. Each project represents one AP application. Each project is represented in Project-Manager window by a project name and an image. You may change project name and its image as shown by FIG. 12.

These contents are not related to action list building. My action list building related contents are in step D, D1, D1a, D1b, and D1c. These steps leave no room for computer coding and using of computer languages. Every step is an interactive operation between

my system and the user. All these steps in my claims put together show and teach a codeless programming process.
The examiner simply says "which does not **appear** to be codeless...the two **appear** to conflict", but coding can be found nowhere in my claims. Such a rejection reason is totally subjective and groundless.

Let's examine the references the examiner used to make the rejections.

1) Budd page 91 and page 92. The examiner says: "the features are taught by Budd to enable interfaces and user interactions, see page 91 (the actionPerformed and actionListener methods, the FirebuttonListener class on page 92 and the data that supports each)". Here the only similarity of these contents in Budd page 91 and 92 is the word "action". The contents are totally different than what I defined in step A of my claims for my action class and how my action class is used as described in all my other claims. I am not patenting the English word "Action". Budd is teaching Java coding: how to write Java code. My claims are teaching a new codeless programming process. The examiner's reference has no relation to my invention. The examiner's understanding of his reference is also questionable because "interface", as taught in these pages, has nothing to do with "user interactions". "interface" is a very specific computer language term referring to a contract between software components, not related to user actions.
2) Budd page 89. The examiner says: "to enable dynamic user interactions, see the last two paragraphs of page 89". Here Budd is teaching a sample application to enable dynamic user interactions, not to build a programming system. Budd's sample application fires cannon. Use a variable to dynamically control the angle of firing: that is what Budd calls "dynamic user interactions". Almost all computer applications have such "dynamic user interactions". It has nothing to do with how any programming systems are working, less my codeless programming system. I am not patenting "dynamic user interactions".
3) Budd page 227 – 231. The examiner says: "The action list class is considered taught via Budds menu items on page 227-231". My action list contains instances of my action class. My action class is defined in step A of my claims, which is totally different from a menu item. It is totally groundless to consider my action list to be the same as menu items. The examiner thinks my action list is used for displaying purpose and thus functions the same as menu item, and thus is not patentable. But my action list is not for displaying purpose. It is for executing actions in a sequence according to the programming needs of the programmer. Menus are for displaying at runtime, and only execute one selected menu item. My action list is built at design time as a programming process. My action list is not displayed at runtime. All actions in my action list are executed when my action list is executed. Comparing my action list to menus is to compare orange with apple. This argument was already presented to the examiner in my response to the first rejection. In the second (final) rejection the examiner re-uses his rejection reasons from the first rejection but refers nothing to my original arguments to his first rejection, as re-presented above. I do not believe it is true

that the examiner fully considered my responses the first time, even the examiner says so in his final rejection.

Argument 8.
Regarding office action the second paragraph of page 8
Rejection reasons to be argued: claim for "an object" is not patentable.
Argument details:
This is another example of "off target" rejection. I am not patenting "an object".

In software engineering, compilation means transferring contents from one format(s), usually in plain text, to another format(s), usually in binary executable format. Claim 6 describes the process of saving, loading and executing the programming results done by the user. As shown in steps F, G, H, H1, H2, H3, H3a, H3b, H3c, H3c1, H3c2, H3c3, and H3c4, the execution is done using the objects created in the process described in claim 1, 2, and 3. Therefore there is not a content format conversion involved. In another word, there is not a compiling involved. Thus my codeless programming system works without using a compiler.

Argument 9.
Regarding office action the last paragraph of page 11
Rejection reasons to be argued: claim is unclear.
Argument details:
If the examiner cannot correctly understand the contents of his own references and shows lack of common sense in software engineering, as above proved, and he never talked to me about my application to let me explain it, I do expect he cannot understand lots of things in my application.

All other rejection reasons not cited above are "off target" rejections. I am not repeat them here.

JDW AF

To: USPTO

From: David Ge
10218 125th AVE NE
Kirkland WA 98033
USA
dge893@gmail.com
(425)803-0679

Date: Friday, July 20, 2007

RE: Complaints about the processing of my application (application #: 09/682,315)

To Whom It May Concern:

USPTO is a respectful office. I hope what I reported may help improve its services to tax payers.

1. My responses to office actions were ignored several times. I have US Post Office receipts to support this complaint.
    a. The first office action I got from USPTO after 3 years and 2 months from the filing date was on 10/01/2004. My response mailed in December, 2004. USPTO received my response on DEC 27 2004, according to US Post Office receipt. In the next more than half a year, I had been inquiring about my response, both by phone and by mails. During this process what I was experiencing was ignorance and irresponsive from USPTO.
    b. On 01-31-2005 USPTO logged my inquiry letter. But I did not get any responses to what I was inquiring.
    c. I made phone calls to the examiner, "CHAVIS, JOHN Q", to push him look into the matter. He asked me to prove that I sent my response in time. I faxed over my receipt from United States Post Office to prove that I had mailed my response in time. Finally he asked me to fax my whole response over, including all drawings prepared for me by a patent drawing company, more than 40 pages.
    d. On 03-22-2005 USPTO logged docketing of my response. It could be the logging of my faxing.
    e. Then it came a strange logging in the USPTO web. An entry of receiving my response on December 27 2004 was logged after the March 22 2005 log entry.
    f. A stranger log entry was entered after the above strange logging. Note the entry of "12-27-2004 Informal or Non-Responsive Amendment after Examiner Action". It tells us that while USPTO accepted the fact that I did mail out my response on DEC 27 2004, it immediately decided that my response was "Informal or Non-Responsive Amendment after Examiner Action". It was logged on the same day, 12-27-2004, as the logging of receiving my response.

g. That immediate decided action, "Informal or Non-Responsive Amendment after Examiner Action", was hold until on 07-18-2005 to be mailed out to me.
h. In March 2006 I sent to USPTO my response to the office action requiring me of election/restriction of my claims. But near the end of 2006, I heard nothing from USPTO. I made phone calls to the examiner, "CHAVIS, JOHN Q". He said: "your application was abandoned because you did not respond in time. You can appeal if you want". I told him I already sent my response long time ago. He said that they did not receive it and told me that I should have included a document list when sending a response.
i. I called the examiner, "CHAVIS, JOHN Q", again later and asked him the proper channel I could file a complaint. He asked me what I had to complaint. I told him I wanted to complain ignoring my mails because I had USPTO receipts for the ignored mails. Mr. CHAVIS told me not file a complaint. He would send me another office notice to let me respond to it and he would allow the patent.
j. Mr. CHAVIS did send me an office notice, although the notice looks like he had a phone conversation with me about my application while actually the phone conversations were really never about the application contents. I responded it.
k. But Mr. CHAVIS lied to me. Instead of allowing my application as he told me on the phone, he sent me a final rejection based on a reference which he misunderstood. His misunderstood of his own referenced materials is a proof of his lack of basic software engineering qualifications to be an examiner for software applications.
l. I immediately filed a Notice of Appeal at the end of April. USPTO logged it on 05-04-2007. I was waiting for the response from the board of appeal so that I can send in my appeal brief. I hoped to establish a contact other than Mr. CHAVIS so that my mails would not be ignored again.
m. Just before the Christmas 2007, Mr. CHAVIS called while I was on vacation out of the town. The person answered the phone told Mr. CHAVIS that I was on vacation and could not be reached, and told him I would call him after Christmas. I tried to reach Mr. CHAVIS and left message asking what was the purpose of his call. He never called me again. Instead he simply sent me an "Abandonment for Failure to Respond to Office Action" on the day of the Christmas 2007.

2. An USPTO examiner could lie to me was completely out of my imagination. When it happened it gave me a big surprise and blow. He ignored my response for almost a year. When I called he said my application was abandoned because of not responding. When I said I had proof of sending my response and I was going to file a complaint, he changed his mind and told me not file a complaint and he would send a new office notice and I would respond it and he would allow it. I agreed. But instead of allowing it, he gave me a final rejection using groundless reasons. I still believe the majority of the USPTO staff is honest. But this examiner certainly has low moral.

3. Another surprise to me is how incompetent an USPTO examiner can be. As evidences in the examiner's rejection notice show that Mr. CHAVIS misunderstood fundamentally his own referenced materials, and lacked of common sense in software engineering. See my appeal brief for explanations of those evidences.
4. The way the examiner processed my case is careless and inefficient. The examination started after 3 years of filing. Since the examination started, basically the case was touched by the examiner once every year. In 2004, the office notice was about the text formatting issues of claim amendment. In 2005, the office notice was about election of claims. In 2006 it was about the drawings. Why not put all the issues in one office action? The examiner's way of working was to pick something wrong, send a notice and put the case away for another year, or even forever by ignoring the responding mails.
5. I was hoping to establish a contact other than Mr. CHAVIS when I filed my Notice of Appeal, and then send in my appeal brief. According to USPTO web, "(f) If a notice of appeal or cross appeal is timely filed but does not comply with any requirement of this section, appellant will be notified of the reasons for non-compliance and given a non-extendable time period within which to file an amended notice of appeal or cross appeal.". But now Mr. CHAVIS terminated my case. Due to low moral and lack of professional qualifications exhibited by the examiner, my case has not been fairly and efficiently processed since its filing on 08-18-2001. I request USPTO to investigate it. I will also publish it to media and internet.